DSpace Hibernate-Prototype

Andrea Bollini, Daniele Ninfo
CILEA, Roma

Abstract
In questo articolo viene illustrata la reingegnerizzazione dello strato di persistenza del software open source DSpace, allo scopo di introdurre un maggiore disaccoppiamento dei layer software. Il prototipo è stato realizzato sfruttando design patterns e standard propri del mondo J2EE: Domain Store, Session Façade, Application Service, JPA e si propone quindi anche come un aggiornamento tecnologico della piattaforma. L'attività di ricerca e sviluppo finalizzata a questo obiettivo è stata condotta nell'ambito di uno stage offerto dal CILEA per il conseguimento della laurea magistrale in Ingegneria Informatica, opportunità che CILEA offre per diverse tematiche ICT all'avanguardia e sotto la supervisione dei propri esperti.

This article shows the reengineering of the persistence tier of the open source software DSpace, to introduce a lower coupling in its software layers. The prototype was made using design patterns and standards from J2EE world: Domain Store, Session Façade, Application Service, JPA were also introduced for a technological upgrade of the platform. The research and development activity aimed at this goal was accomplished in a stage offered by CILEA to achieve a degree in Computer Science Engineering, an opportunity that CILEA offers on several innovative ICT themes, with the support of its experts.

Keywords: DSpace, Hibernate, reengineering, JPA, digital library, design pattern.

Introduzione e contesto

DSpace è una piattaforma open source per la creazione e la gestione di archivi aperti, che consente di archiviare dati in diversi formati, di renderli fruibili attraverso la rete, di indicizzarli e preservarli [1]. Offre la possibilità di gestire pubblicazioni di ricerca o materiali per la didattica in un repository che garantisce ai dati visibilità e accessibilità nel tempo, in modalità compatibile con le specifiche del protocollo per l'interoperabilità OAI-PMH [2]. Il CILEA, nell'ambito della gestione di biblioteche elettroniche, fornisce servizi di installazione e personalizzazione di questo software, assumendo una posizione di leadership in Italia nell'offerta di consulenza grazie all'elevato numero di installazioni effettuate della piattaforma e della stretta collaborazione mantenuta con la community open source. Grazie al know how acquisito, il CILEA può offrire soluzioni altamente scalabili e un ampio spettro di servizi aggiuntivi ad elevato valore aggiunto.

La possibilità di effettuare economie di scala consente inoltre di offrire ai nostri clienti e consorziati soluzioni che, se sviluppate direttamente o acquisite da fornitori meno specializzati, avrebbero costi proibitivi. In particolare il CILEA ha selezionato DSpace come fondamenta del modulo Open Archive [3] della propria suite applicativa SURplus [4], strumento di controllo e monitoraggio per atenei e centri di ricerca a supporto della valutazione e disseminazione dei risultati della ricerca. La forte aderenza a standard internazionalmente riconosciuti e l'utilizzo di una tecnologia robusta ed affidabile come J2EE ne semplificano infatti l'integrazione ed estensione, anche quando questi non possono essere condivisi con l'intera comunità essendo magari limitate a specifiche esigenze locali/nazionali, come ad esempio il dialogo con database ministeriali come il "Sito docente".

Progettato per lavorare "out of the box", svolgendo azioni di archiviazione, rimanendo personalizzabile e gestendo la preservazione nel tempo dei contenuti secondo lo standard OAIS, DSpace è stato sfruttato in vari ambiti, diffondendosi in larga scala presso varie istituzioni. Il principale contesto di utilizzo è comunque l'editoria elettronica, in particolare presso diverse università: la funzione che assume è

quella di archiviare documenti quali tesi di laurea, articoli scientifici, pubblicazioni varie.

Vista la larga diffusione, è necessario un continuo adeguamento per supportare la diversità di applicazione che DSpace si trova a gestire, mantenendo sempre attivo l'obiettivo della preservazione a lungo termine. Alcuni sviluppi possono essere realizzati semplicemente scrivendo delle patch o reimplementando componenti specifiche, ma è necessario rivedere periodicamente l'architettura dell'intero sistema per essere sicuri che possa venire incontro alle necessità dei sempre più numerosi utilizzatori. Per questo l'architettura software di DSpace è in continua evoluzione, grazie ai contributi di una comunità di sviluppatori che lavora attivamente, coordinata da un gruppo di Committers [5] definito sulla base di criteri meritocratici. Il CILEA è presente con un suo esperto all'interno di questo gruppo di sviluppatori di alto livello. Questa importante presenza garantisce un ruolo rilevante negli sviluppi innovativi della piattaforma.

Il progetto Hibernate-Prototype è inquadrato in questo contesto di evoluzione continua, nel quale le soluzioni vengono proposte da diverse fonti, valutate ed eventualmente integrate, senza però escludere esperimenti e personalizzazioni nello sviluppo e nell'utilizzo: la natura open source della piattaforma consente un adattamento del prodotto alle proprie esigenze, con una successiva condivisione dei risultati.

Scopo del progetto era la ristrutturazione di alcune parti dell'architettura di DSpace, che potessero portare ad un prototipo che vedesse la realizzazione di soluzioni alternative a diversi problemi, da proporre poi alla comunità di sviluppo per una possibile integrazione di queste nelle versioni future dell'applicazione.

L'architettura di DSpace è strutturata secondo tre livelli principali:

- lo Storage Layer è responsabile della gestione dei dati persistenti, in particolare interagisce con un DBMS.
- Il Business Logic Layer realizza la logica di controllo per la manipolazione dei dati e offre servizi attraverso delle API, che vengono sfruttate dal livello superiore.
- L'Application Layer è formato da moduli che costituiscono le interfacce per l'interazione con gli utenti finali.

DAO Prototype

Nelle versioni che si sono succedute di DSpace sono evolute anche le modalità di realizzazione di questi tre livelli, con l'introduzione di metodologie e tecnologie diverse. Il progetto descritto nell'articolo si è sviluppato temporalmente tra settembre 2007 e marzo 2008, cioè nel passaggio dalla versione 1.4 alla versione 1.5 (quest'ultima è l'attuale versione, scaricabile dal sito [6]), ma ha avuto come base un prototipo che propone un rifacimento architetturale dello strato di persistenza, con l'introduzione del pattern DAO: seguendo questo pattern, il prototipo fa uso di uno strato software di oggetti responsabili di rendere persistenti le modifiche ai dati. Il prototipo DAO [7] [fig. 1] è stato scelto come base per lo sviluppo della futura versione 2.0, ma non si esclude una sua introduzione in versioni intermedie, come ad esempio la 1.6.

La realizzazione del prototipo DAO tuttavia, pur rappresentando un notevole miglioramento dello strato di persistenza, presenta comunque a nostro avviso alcuni problemi architetturali, dei quali quello a più alto livello è l'elevato accoppiamento tra il Business Logic Layer e lo Storage Layer: gli oggetti DAO, strato software dello Storage Layer, realizzano parte della logica di creazione e sono utilizzati attivamente dagli oggetti del modello, per questo il livello di business è fortemente dipendente da quello sottostante. Ciò è dovuto a una distribuzione non corretta delle responsabilità: un DAO dovrebbe avere unicamente il ruolo di interagire con lo strato persistente (in questo caso il DB), non dovrebbe essere parte attiva per la realizzazione della logica; viene creata una dipendenza non necessaria che riduce l'indipendenza dei livelli. Da ciò consegue una frammentazione della logica che, oltre ad essere realizzata da oggetti del modello e da alcuni manager, è realizzata da elementi dello Storage Layer, che diventa inoltre visibile allo strato di presentazione: i moduli dell'Application Layer, per interagire con le API di business, utilizzano sia oggetti del Business Logic Layer che oggetti DAO, in questo modo la struttura interna del livello più basso è nota al livello superiore, in contrasto quindi con il principio di incapsulamento.

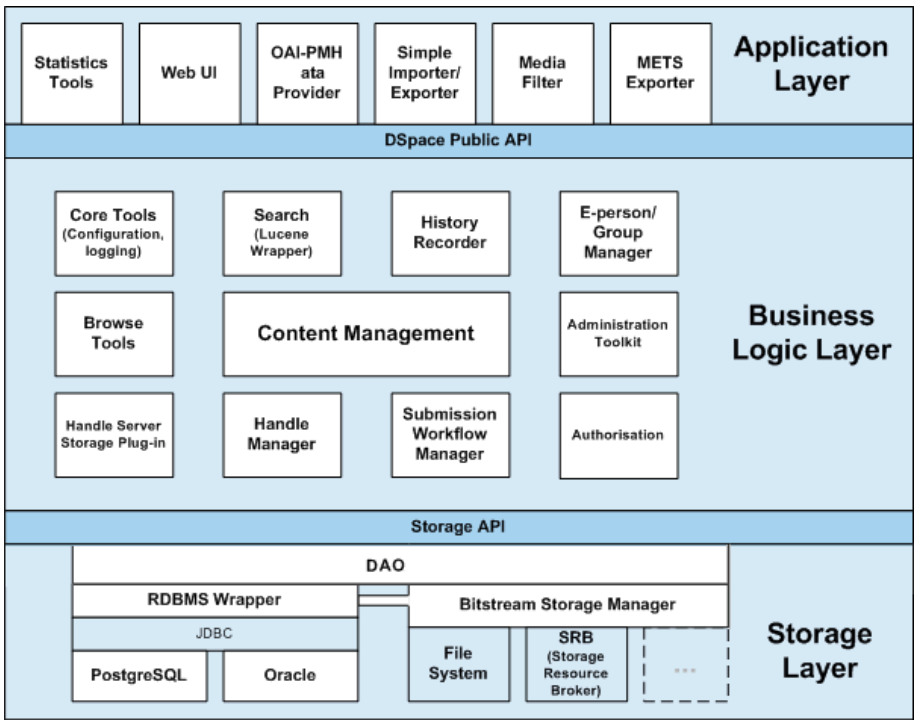

Fig. 1 - L'architettura del prototipo "DAO"

Hibernate Prototype

La ristrutturazione proposta ha avuto come obiettivo la soluzione di questi problemi, mediante l'utilizzo di design pattern noti e di tecnologie che potessero essere di supporto allo sviluppo. Le soluzioni proposte hanno visto un approccio Domain-Driven Design, seguendo il quale sono stati revisionati i livelli di business e di persistenza.

Per la revisione del Business Logic Layer sono stati seguiti due pattern J2EE [8], Session Façade e Application Service. Gli oggetti del modello di Hibernate-Prototype presentano ora solo attributi, senza comportamento: gli unici metodi sono di tipo getter/setter. La logica è stata spostata interamente su dei manager appositi, sollevando da questa responsabilità sia gli oggetti del modello che gli oggetti DAO: i metodi dei manager costituiscono ora le API dello strato di business, che vengono sfruttate dal livello applicativo. I manager potrebbero essere inoltre a loro volta mascherati da una session façade che si può proporre come punto di accesso centralizzato al Business Logic Layer.

Lo Storage Layer è stato revisionato sfruttando il pattern Domain Store. Il livello è totalmente trasparente ai livelli superiori, che ne conoscono solo l'interfaccia: l'intero strato software è infatti stato mascherato da una classe apposita, che offre agli altri manager del livello di business servizi di persistenza. Per implementare il pattern è stata utilizzata lo standard Java Persistence (JPA), con Hibernate [9] come JPA-Engine. Tutti gli oggetti del modello sono stati mappati con le JPA Annotations: basandosi su questo mapping è ad esempio possibile generare automaticamente lo schema del DB. Il punto di forza di questa tecnologia sta nella possibilità di cambiare facilmente JPA-Engine, passando ad esempio dalla libreria Hibernate [9] di JBoss alla libreria TopLink [10] della Oracle. Mentre le operazioni di persistenza più semplici (CRUD) vengono affidate al JPA-Engine attraverso delle interfacce, le operazioni che richiedono interrogazioni più complesse verso il DB sono responsabilità degli oggetti DAO, che esonerati dalla realizzazione della logica, hanno nel nostro prototipo solo la funzione di interazione con il DB, come dettato dal pattern DAO. Lo standard JPA introduce inoltre un nuovo linguaggio di interrogazione: JPA QL che consente di utilizzare logiche proprie della programmazione ad oggetti nell'interrogazione dei dati, lasciando al JPA Engine il compito di risolvere i problemi di impedenza propri dell'Object Relational Mapping.

L'utilizzo di JPA consente dunque di essere svincolati dal particolare DBMS utilizzato: le release ufficiali di DSpace hanno già un certo

grado di indipendenza consentendo l'utilizzo di Postgres o Oracle ma, di fatto, sono dipendenti fortemente da questi. Ogni nuovo sviluppo può compromettere la compatibilità con uno dei due DBMS, questo richiede un'attenta attività di testing e un rallentamento nella condivisione degli sviluppi quando questi non siano automaticamente portabili da un ambiente ad un altro. Inoltre, nonostante Postgres e Oracle rappresentino forti punti di riferimento nei rispettivi mercati (open source e commerciale) non si può escludere l'opportunità di utilizzare, in particolare contesti, anche solo per omogeneità con un'infrastruttura pre-esistente, altri DBMS quali ad esempio MySQL o SQL Server.

Con JPA si ha una tecnologia totalmente indipendente dallo strato di persistenza sottostante e si demandano i problemi di portabilità a specifiche comunità (Hibernate, Toplink, etc.) consentendo quindi agli sviluppatori DSpace di concentrarsi solo sulle funzionalità proprie dell'applicativo.

Con queste ristrutturazioni si è arrivati ad una architettura alternativa di DSpace (fig. 2), con il prototipo liberamente accessibile nella sandbox di DSpace [11] e documentato sul wiki [12], offerto come veicolo di valutazione delle scelte tecnologiche e metodologiche da parte dei Committers e dell'intera Community.

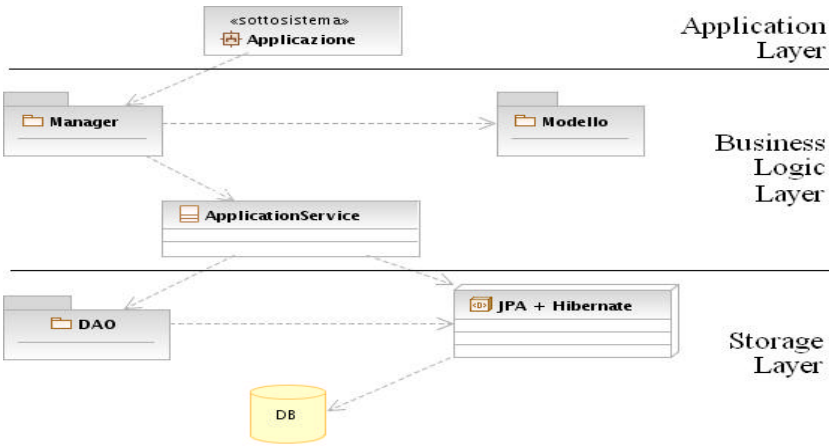

Fig. 2 - L'architettura del prototipi "Hiber Nate"

Bibliografia

[1] Bollini, A. & Meschini, F. (October, 2007). DSpace Basic Tutorial. Paper presented at DSpace User Group Meeting 2007, Food and Agriculture Organization of the United Nations, Rome, Italy – URL:http://www.aepic.it/conf/viewabstract.php?id=321&cf=11

2] The Open Archives Initiative Protocol for Metadata Harvesting. URL:http://www.openarchives.org/OAI/openarchivesprotocol.html

[3] Mornati, S., & Bollini, A. (October, 2007). New opportunities for Institutional Repositories: the evaluation challenge. A case study. Paper presented at DSpace User Group Meeting 2007, Food and Agriculture Organization of the United Nations, Rome Italy.URL:http://www.aepic.it/conf/viewabstract.php?id=214&cf=11

[4] SURplus URL: http://www.cilea.it/index.php?id=SURplus

[5] DSpace Committers & Contributors URL: http://wiki.dspace.org/index.php/DspaceContributors

[6] URL: http://www.dspace.org

[7] DAO-Prototype on DSpace Wiki URL: http://wiki.dspace.org/index.php/DAO_Prototype

[8] Core J2EE Patterns: Best Practices and Design Strategies. Deepak Alur, John Crupi and Dan Malks (2nd Edition), URL: http://www.corej2eepatterns.com/

[9] Hibernate web site. URL: http://www.hibernate.org

[10] Toplink web site. URL:http://www.oracle.com/technology/products/ias/toplink/index.html

[11] SVN Repository. URL:http://code.google.com/p/dspace-sandbox/source/checkout

[12] Hibernate-Prototype on DSpace Wiki. URL:http://wiki.dspace.org/index.php/Hibernate_prototype